# Building Linux for the ADMXRC7Z1/2

# Introduction

This guide describes how to build the Open Source Embedded Linux distribution "Poky" using the Yocto build system, and "Das U-Boot", for an ADM-XRC-7Z1 or ADM-XRC-7Z2.

# Building Linux Kernel and Root File System

## Required Tools

To follow this procedure you will require a PC or Virtual Machine running a Desktop Linux distribution. See the 'The Linux Distribution' section in http://www.yoctoproject.org for instructions on setting up the required packages on your Linux Desktop system.

## Sources

After setting up a Linux desktop environment for the Yocto build system, **git** can be used to acquire the source code required. Start by creating a new directory for your Embedded Linux project. From inside the new directory use the following commands to acquire a copies of the required code bases.

```
git clone -b fido git://git.yoctoproject.org/poky.git
git clone -b fido https://github.com/Xilinx/meta-xilinx.git
git clone -b fido https://github.com/openembedded/openembedded-core.git
git clone -b master https://github.com/adps/meta-adlnx.git
git clone -b master https://github.com/adps/meta-admxrc7z.git
```

The last two code bases, **meta-adlnx** and **meta-admxrc7z**, are Alpha Data's example root files system, and Alpha Data's board support package for the ADM-XRC-7Z1 and ADM-XRC-7Z2.

## Building

From the **~/poky** sub-folder in your Embedded Linux project directory, enter the following command to initialise the build environment:

```
source oe-init-build-env
```

This will create a build directory, **build**, which will be switched into to as the active directory after the script completes execution.

### Build setup

Before a build can be started, two files must be edited in the **~/poky/build/conf** directory; poky/build/conf/bblayers, and poky/build/conf/local.config

### poky/build/conf/bblayers

The **bblayers** file must be modified to include the full path of additional layers that need to be added to the Yocto Linux build system. Edit **bblayers** to be similar to the following, including the **meta-adlnx** and **meta-admxrc7z**

and layers and the **openembedded-core/meta** layer. Note the path **/home/my_home/yocto_linux** should be changed to the full path of your Embedded Linux project directory.

```
# LAYER_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
LCONF_VERSION = "6"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
/home/my_home/yocto_linux/openembedded-core/meta \
/home/my_home/yocto_linux/meta-xilinx \
/home/my_home/yocto_linux/meta-adlnx \
/home/my_home/yocto_linux/meta-admxrc7z \
/home/my_home/yocto_linux/poky/meta \
/home/my_home/yocto_linux/poky/meta-yocto \
/home/my_home/yocto_linux/poky/meta-yocto-bsp \
  "
BBLAYERS_NON_REMOVABLE ?= " \
  /home/my_home/yocto_linux/poky/meta \
  /home/my_home/yocto_linux/poky/meta-yocto \
  "
```

## poky/build/conf/local.config

The **local.config** file must be modified to specify the target machine. Edit this file to include the following line to select the target machine:

```
machine ?= admxrc7z
```

## Build

Use the following command in the **~/poky/build directory** to start a build of the Embedded Linux Kernel and root file system. This will take some time to complete.

```
bitbake adlnx-image
```

The output should look similar to the following:

```
Loading cache: 100% |#####################################################################
##########################################################| ETA:  00:00:00
Loaded 2592 entries from dependency cache.
Parsing recipes: 100% |###################################################################
##########################################################| Time: 00:00:01
Parsing of 1787 .bb files complete (1784 cached, 3 parsed). 2591 targets, 128 skip
ped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION        = "1.26.0"
BUILD_SYS         = "x86_64-linux"
NATIVELSBSTRING   = "openSUSE-project-13.2"
TARGET_SYS        = "arm-poky-linux-gnueabi"
MACHINE           = "admxrc7z"
DISTRO            = "poky"
DISTRO_VERSION    = "1.8"
TUNE_FEATURES     = "arm armv7a vfp neon cortexa9"
TARGET_FPU        = "vfp-neon"
meta
```

```
meta-yocto        = "fido:f366ff2c03885f0ac17415dfbc8f25b2b760b841"
meta              = "fido:cd3da9c95f48899e134a5b7ed1754fd18985df4f"
meta-xilinx       = "fido:276717514962728501de0b69aee0d9052d9a1366"
meta-adlnx        = "master:107fff621ea6ef730c9bc6644800a976cc66b482"
meta-admxrc7z     = "master:a36ad52e7fb72fc30e43b9fe0b4259f22c8d4767"

NOTE: Preparing runqueue
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
```

## Output files

After the build is completed, several output files are created in **poky/build/tmp/deploy/images/admxrc7z/**

**uImage-admxrc7z1.bin**

The Linux Yocto Poky Daisy kernel image. This is the core of the operating system, but requires a device tree to provide details of the hardware configuration.

**uImage-admxrc7z1.dtb**

Device tree blob. This provides details of the hardware configuration to the kernel.

**core-image-minimal-admxrc7z1.cpio.gz.u-boot**

The root file system as a RAM disk containing a CPIO image prepended with a Das U-Boot header.

**core-image-minimal-admxrc7z1.ext2**

The root file system as an ext2 binary image.

The Embedded Linux kernel can use either the CPIO version of root file system or the ext2 version as its root file system. The kernel boot arguments determine the format and device on which the root file system resides. For example, the ext2 version of the root file system could be written to an SD card and Das U-Boot can direct the kernel to boot from the appropriate partition of the SD card.

## Building a standalone toolchain

In addition to building the root file system and Kernel a standalone toolchain might be needed for developing application outside the Yocto enviroment. This can be build with the following command.

```
bitbake adlnx-image -c -c populate_sdk
```

Output files for this consist of a poky-glibc-x86_64-adlnx-image-cortexa9-vfp-neon-toolchain-1.8.sh installer created in **poky/build/tmp/deploy/sdk/**. Running this on a Linux system will allow the user to install cross compiler tools customised to target the adlnx reference image.

# Emulation with QEMU

To build an image that will run on the QEMU virtual machine name in the local.conf file to **qemuzynq**. After modifying the local.conf file use **bitbake adlnx-image** to build a kernel and root file system that will be compatable with the QEMU.

To run the QEMU with the built image execute the following command

```
runqemu qemuzynq nographic qemuparams="-m 512"
```

# Das U-Boot

To make it easier to develop and update Embedded Linux images, the FSBL (First Stage Boot Loader) will load Das U-Boot. Das U-Boot is a second stage boot loader that allows a number of boot options, including booting from QSPI, SD card, and TFTP.

## Sources

Acquire the sources to build Das U-Boot using the following command:

```
git clone -b master https://github.com/adps/u-boot-ad-zynq7.git
```

## Building Das U-Boot

Execute the following commands inside the u-boot-ad-zynq7 directory:

```
export CROSS_COMPILE=arm-xilinx-linux-gnueabi-

source /opt/Xilinx/Vivado/2013.4/settings64.sh

export BUILD_DIR=build

make admxrc7z1_config

make all
```

### Output files

After building has completed, the following files can be found in the build directory:

| | |
|---|---|
| **u-boot.elf** | This file is an ELF version of the built U-Boot binary. |
| **BOOT.bin** | This file is a bootable image containing both a basic FSBL for the ADM-XRC-7Z1 and the U-Boot binary. |

# Testing

The default environment variables in Das U-Boot will attempt to boot the **uImage-admxrc7z1.bin** kernel image with the **core-image-minimal-admxrc7z1.cpio.gz.u-boot** root file system and **uImage-admxrc7z1.dtb** device tree.

## Preparing an SDCard

To test the built Das U-Boot and Embedded Linux image, follow this procedure:

1. Configure the ADM-XRC-7Z1 for boot from uSD, by setting SW2-2 to On and SW2-1 to off.
2. With the ADM-XRC-7Z1 powered down, install the uSD card in the uSD card slot.
3. Depending on the carrier and break out board being used, connect a serial cable to the S0 serial output from the ADM-XRC-7Z1. See the User Manual for the ADM-XRC-7Z1 for details of switches and connectors.
4. After powering on the ADM-XRC-7Z1 it should boot from the uSD card, first showing information about the ADM-XRC-7Z1 in the FSBL, before proceeding to load and execute U-Boot. U-Boot will wait for three seconds for a user to intervene before starting to load Embedded Linux.

After Linux has booted, the user name **root** can be used to log into the system (without a password).

# Typical Boot Output

```
             @@@@@@@
          @@@@@@@@@@@@
        @@@@@@@@@@@@@@@@
       @@@@@@@@@@@@@ @@@
     @@@@@@@@@@@@@@ @@@@
    @@@@@@@@@@@@@@ @@@@@
  @@@       @@@@@ @@@@@
 @@ @@@@@@@@ @@ @@@@@
 @ @@@@@@@@@@  @@@@@    @    @@    @@@  @  @    @       @@@@     @    @@@@    @
 @@  @@@@@@@ @@ @@     @ @   @@    @@ @ @@@@  @ @      @@ @@   @ @    @@    @ @
 @@@@        @@@@      @@@@@ @@    @@@  @@@@ @@@@@     @@ @@ @@@@@   @@   @@@@@
  @@@@@@@@@@@@         @@ @@ @@@@ @@    @  @ @@ @@     @@@@  @   @@   @@   @@ @@
     @@@@@@
```

```
Alpha Data Parallel Systems Ltd. Zynq7 FSBL Release 1.1.0.0

Board is in application mode. System monitor reporting:

 Brdman Ver : 1.1.1.0

 Board Type : ADM-XRC-7Z1       Serial Number : 102
         P6 : Fitted                       P4 : Fitted
  Micro-USB : Fitted                  Micro-SD : Fitted
   Prog Clk : 150000000Hz          Tgt Stepping :
Speed Grade : 1                     Tgt Device : 7Z045
 Temp Grade : Industrial                   PCB : 1.0
       CPLD : 1.0                     PS SDRAM : 512MiB@800.0MHz
      QSPI0 : 32MiB                      QSPI1 : 32MiB
  PL SDRAM0 : 256MiB@800.0MHz       PL SDRAM1 : 256MiB@800.0MHz

       VPWR :  11.837V (OK)          12V0_DIG :  11.816V (OK)
    5V0_DIG :   5.028V (OK)           3V3_DIG :   3.264V (OK)
    2V5_DIG :   2.494V (OK)        AUX_IO_2V0 :   1.980V (OK)
    1V8_DIG :   1.790V (OK)       MGT_AUX_1V8 :   1.819V (OK)
   VCC_DRAM :   1.497V (OK)           XRM_VIO :   0.000V (OK)
    1V0_DIG :   0.994V (OK)           TGT_1V2 :   1.196V (OK)
    TGT_1V0 :   0.990V (OK)        3V3_DIG_IC :   3.260V (OK)
   AVR Temp :  32.350C (OK)          PCB Temp :  32.725C (OK)
Target Temp :  31.475C (OK)       Alarm State : OK

   PSU_OK_0 : TRUE                   MezPres0 : FALSE
  FPGADone0 : TRUE                Force 2V5 0 : FALSE
 PCIe Reset : TRUE                XRM Present : FALSE
 XRM VIO En : FALSE                XRM I2C Er : FALSE
XRM Chks Er : FALSE               XRM Data Er : FALSE
  Force 2V5 : FALSE                 SI5338 OK : TRUE
 Intrrgt OK : FALSE                   Op 1 OK : FALSE
         EC : 811                         ETC : 19d:7h:31m:0s

Booting from SD card ...
```

The last line of the above output represents the point at which Alpha Data's FSBL hands over control to Das U-Boot. Note: There may also be some additional debug output from the FSBL, if it is compiled for debug.

Following the FSBL output is output from Das U-Boot, which takes the following form:

```
U-Boot 2014.01-g67f6167-dirty (Aug 11 2014 - 15:28:55)

I2C:   ready
Memory: ECC disabled
DRAM:  512 MiB
MMC:   zynq_sdhci: 0
SF: Detected N25Q256A with page size 256 Bytes, erase size 4 KiB, total 64 MiB
In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
Hit any key to stop autoboot:  0
Device: zynq_sdhci
Manufacturer ID: 2
OEM: 544d
Name: SA08G
Tran Speed: 50000000
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 7.3 GiB
Bus Width: 4-bit
reading uEnv.txt
** Unable to read file uEnv.txt **
Copying Linux from SD to RAM...
reading uImage-admxrc7z1.bin
3578392 bytes read in 324 ms (10.5 MiB/s)
reading uImage-admxrc7z1.dtb
23630 bytes read in 20 ms (1.1 MiB/s)
reading core-image-minimal-admxrc7z1.cpio.gz.u-boot
2419107 bytes read in 224 ms (10.3 MiB/s)
## Booting kernel from Legacy Image at 03000000 ...
   Image Name:   Linux-3.14.2-xilinx
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3578328 Bytes = 3.4 MiB
   Load Address: 00008000
   Entry Point:  00008000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 02000000 ...
   Image Name:   core-image-minimal-admxrc7z1-201
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    2419043 Bytes = 2.3 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 02a00000
   Booting using the fdt blob at 0x2a00000
   Loading Kernel Image ... OK
   Loading Ramdisk to 1f8e1000, end 1fb2f963 ... OK
   Loading Device Tree to 1f8d8000, end 1f8e0c4d ... OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 3.14.2-xilinx (dd@dd4-linux) (gcc version 4.8.2 (GCC)
) #1 SMP PREEMPT Wed Aug 6 15:45:04 BST 2014
[    0.000000] CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=18c5387d
```

```
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[     0.000000] Machine model: Xilinx Zynq
```

The above output ends with Das U-Boot handing over execution to the Embedded Linux kernel. As execution continues various debug information will be printed reporting the status of drivers being loaded and the operating system being configured.

The boot stage of Embedded Linux will end logging similar output to that shown below.

```
Mon Aug  4 16:01:00 UTC 2014
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc (v1.22.1) started
Sending discover...
Sending discover...
[   11.255343] macb e000b000.ps7-ethernet eth0: link up (1000/Full)
Sending discover...
Sending select for 192.168.2.70...
Lease of 192.168.2.70 obtained, lease time 691200
/etc/udhcpc.d/50default: Adding DNS 192.168.2.15
/etc/udhcpc.d/50default: Adding DNS 192.168.2.8
done.
Starting syslogd/klogd: done
Stopping Bootlog daemon: bootlogd.

Poky (Yocto Project Reference Distro) 1.6.1 admxrc7z1 /dev/ttyPS0

admxrc7z1 login:
```

# Extending and developing

Alpha Data's board support meta layer (**meta-ad-zynq7**) is only a basic example. If your FPGA design requires custom drivers and modifications to the device tree, the meta layer can be modified to include these.

# Revision History

| Revision | Date | Description of Change |
|---|---|---|
| 1.0 | 11th August 2014 | Initial draft |
| 2.0 | 14th May 2015 | Updated for using Fido Poky release. |

Address: 4 West Silvermills Lane,
Edinburgh, EH3 5BD, UK
Telephone: +44 131 558 2600
Fax: +44 131 558 2700
email: sales@alpha-data.com
website: http://www.alpha-data.com

Address: 3507 Ringsby Court Suite 105,
Denver, CO 80216
Telephone: (303) 954 8768
Fax: (866) 820 9956 toll free
email: sales@alpha-data.com
website: http://www.alpha-data.com